Bastian Ristau

Entwurfsraumexploration heterogener Multi-Prozessor-Systeme

Beiträge aus der Informationstechnik

Mobile Nachrichtenübertragung

Nr. 53

Bastian Ristau

Entwurfsraumexploration heterogener Multi-Prozessor-Systeme

VOGT

Dresden 2010

Bibliografische Information der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.

Bibliographic Information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibiograpic data is available in the internet at http://dnb.ddb.de.

Zugl.: Dresden, Techn. Univ., Diss., 2010

Die vorliegende Arbeit stimmt mit dem Original der Dissertation
„Entwurfsraumexploration heterogener Multi-Prozessor-Systeme“
von Bastian Ristau überein.

Gesetzt vom Autor

ISBN 978-3-938860-42-7

Jörg Vogt Verlag
Niederwaldstr. 36
01277 Dresden
Germany

Phone: +49-(0)351-31403921
Telefax: +49-(0)351-31403918
e-mail: info@vogtverlag.de
Internet : www.vogtverlag.de

Technische Universität Dresden

Entwurfsraumexploration heterogener Multi-Prozessor-Systeme

Bastian Ristau

von der Fakultät Elektrotechnik und Informationstechnik
der Technischen Universität Dresden

zur Erlangung des akademischen Grades eines

Doktoringenieurs

(Dr.-Ing.)

genehmigte Dissertation

Vorsitzender:	Prof. Dr.-Ing. Eduard Jorswieck
Gutachter:	Prof. Dr.-Ing. Gerhard P. Fettweis
	Prof. Dr.-Ing. Norbert Wehn
Tag der Einreichung:	26.07.2010
Tag der Verteidigung:	03.11.2010

Danksagung

Diese Arbeit steht am Ende einer großartigen Zeit am Vodafone Stiftungslehrstuhl der TU Dresden. Ich durfte auf einem faszinierenden Gebiet arbeiten und dabei sowohl viele neue Erkenntnisse gewinnen als auch interessante Menschen und Orte kennen lernen. Besonderen Dank möchte ich daher an dieser Stelle meinem Doktorvater Gerhard Fettweis aussprechen, der mir dies ermöglicht und dabei stets die notwendigen Freiheiten gelassen hat. Auch danke ich Norbert Wehn sehr herzlich für die Übernahme des Koreferats und seinen damit verbunden hilfreichen Anmerkungen.

Ganz besonderer Dank gilt auch meiner Freundin Katrin, die immer für mich da war und zu mir gehalten hat, auch wenn ich selbst gerade in der letzten Zeit oft nur physisch anwesend war. Ebenso danke ich meinen Eltern, die mich auf dem Weg hierher stets geduldig unterstützt haben, auch wenn dieser an mancher Stelle nicht ganz gradlinig war.

Großer Dank gilt auch meinen Freunden, die immer Verständnis dafür aufgebracht haben, wenn ich gerade mal wieder keine Zeit hatte. Umgekehrt möchte ich Torsten Limberg ebenso dafür danken, dass er immer ein offenes Ohr selbst für meine absurdesten Ideen und Fragen zur Programmierung hatte.

Zusammenfassung

Eingebettete Systeme werden zunehmend als Multi-Prozessor-Systeme umgesetzt. Die wachsende Anzahl parallel zu bearbeitender Applikationen sowie die steigende Komplexität der einzelnen Applikationen selbst führt dabei zu einer stetig steigenden Anzahl der Prozessoren in diesen Systemen, wodurch der zu betrachtende Entwurfsraum nicht-linear anwächst. Mit der Entwurfsraumexploration wird daher versucht, insbesondere folgende Fragen zu beantworten: „*Welche und wie viele Prozessoren sind notwendig, um die aus dem Applikationsszenario resultierenden Anforderungen zu erfüllen?*“ sowie „*Wie kann das gegebene Applikationsszenario sinnvoll auf die parallele Architektur abgebildet werden?*“. Bisherige Ansätze bilden dafür die einzelnen Applikationen manuell oder automatisch auf das zu untersuchende System ab und bestimmen die notwendigen Kennzahlen entweder analytisch oder mithilfe von Simulationen. Die so gewonnenen Ergebnisse werden dann zur Verbesserung einzelner Applikationen, der Architektur oder der Abbildung der Applikationen auf die Architektur verwendet. Bedingt durch die Komplexität des Problems der Bestimmung einer geeigneten Abbildung sind diese Ansätze jedoch zu zeitintensiv als dass sie die Exploration großer Entwurfsräume erlauben. In dieser Arbeit wird daher ein Ansatz vorgestellt, der die globale Exploration über Parallelitätsanalyse ohne die konkrete Abbildung der Applikation auf das System ermöglicht. Erst zur lokalen Optimierung des Systems wird auf die Methode der Abbildung der Applikation auf das System zurückgegriffen. Es wird gezeigt, dass dieses Problem als Packungsproblem aufgefasst und mit linearer Programmierung gelöst werden kann. Der Vergleich des so definierten Referenzpunkts mit der einfachen Heuristik des listenbasierten Schedulings wird zeigen, dass letzteres Vorgehen vergleichsweise gute Ergebnisse ohne das Iterieren über verschiedene Abbildungen liefert. Die geringe Komplexität des Ansatzes erlaubt zudem die Nutzung desselben zur dynamischen Abbildung der Applikationen auf das System zur Laufzeit.

Abstract

Embedded systems are more and more implemented as multi-processor systems. The growing number of applications running in parallel as well as the increasing complexity of the applications result in an increasing number of processors within those systems. Thus, the design space of such systems grows non-linearly. Design space exploration therefore tries to answer the following questions: *"How many processors are necessary to fulfill the requirements defined by the application scenario?"* and *"How can the given application scenario be mapped to the parallel architecture appropriately?"* In existing approaches the applications are either manually or automatically mapped to the system. Performance figures are derived either analytically or by running simulations. The results are then used for modifying single applications or improving either architecture or mapping. Due to the complexity of the mapping problem, current approaches following this methodology are too time consuming to permit the exploration of huge design spaces. In this work an approach is presented, that enables global exploration via parallelism analysis. A mapping is not required until the candidate systems identified in the global exploration are inspected in more detail. It is shown that the mapping problem can be regarded as packing problem and solved with linear programming. The comparison of this reference point with the simple heuristic of list-scheduling will show that the latter produces comparatively good results at low computational complexity. This low complexity also allows for using the heuristic approach for dynamic mapping at run-time.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

2D-SPP	Zweidimensionales Streifenpackungsproblem
ASAP	As Soon As Possible
ASIC	Anwendungsspezifischer integrierter Schaltkreis, engl. Application Specific Integrated Circuit
AST	Abstrakter Syntaxbaum, engl. Abstract Syntax Tree
BDF	Boolean Data Flow, siehe auch [12]
DSE	Entwurfsraumexploration, engl. Design Space Exploration
EDF	Earliest Deadline First
FIFO	First In, First Out
FSM	Endlicher Zustandsautomat, engl. Finite State Machine
KPN	Kahn-Prozess-Netzwerk, siehe auch [44]
LL	Least Laxity. Dabei ist Laxity die Differenz zwischen der Deadline eines Tasks und dessen Ende.
MILP	Gemischt ganzzahliges lineares Programm, engl. Mixed Integer Linear Program
MPSoC	Multi-Processor-System-on-Chip
SDF	Synchronous Dataflow, siehe auch [54]
SSA	Single Static Assignment, siehe auch [22]
TLM	Transaction Level Modeling
UML	Unified Modeling Language, siehe auch [67]

Symbolverzeichnis

$a_{i,j}$	Menge der Daten, die Task i von Task j benötigt.
$b_{i,j}$	Relative horizontale Position der Tasks i und j im 2D-SPP. Dabei ist $b_{i,j} = 1$, falls Task i vor Task j startet.
$C^{\max}$	Miminale Ausführungszeit eines Taskgraphen auf einem System mit unbeschränkten Ressourcen
$d_k(c)$	Durch Prozessortyp k verursachte Verzögerung im Zyklus c
$e(v)$	Spätestmögliches Ende des Tasks v
$f_k(c)$	Parallelitätsprofil des Prozessortyps k im Zyklus c
$f'_k(c')$	Gestrecktes Parallelitätsprofil des Prozessortyps k im neuen Zyklus c'
$f^p_k(c)$	Periodisches Parallelitätsprofil des Prozessortyps k im Zyklus c
$G = (V,E), G_g$	Taskgraph (g), wobei V die Menge der Knoten und E die Menge der Kanten ist
$\overline{G}_g$	Realisierung des Taskgraphen G_g, wobei Realisierung hier die Auswahl eines Pfades bedeutet.
$H^{\max}$	Obere Schranke für die Höhe des Streifens im 2D-SPP.
$L(L^E, L^{\min}, L^{\max})$	(Erwartete, geschätzte minimale und maximale) Laufzeit des Taskgraphen
$m(v)$	Zuordnung des Tasks v auf einen Prozessortyp
p_k	Anzahl der Prozessoren des Typs k
P	Menge der Prozessorentypen
Q	Quellen des Taskgraphen
R	Liste der aktuell schedulebaren Tasks im listenbasierten Mappingansatz
$s(v)$	Frühestmöglicher Start des Tasks v
t	Zeit

t^s	Zeit, die zur Abbildung eines Tasks benötigt wird.
$u_{i,j}$	Relative vertikale Position der Tasks i und j im 2D-SPP. Dabei ist $u_{i,j} = 1$, falls die Nummer des Prozessors, auf den Task i abgebildet wird, größer ist als die des Prozessors von Task j.
$w_{v,k}$	Dauer des Tasks v auf Prozessor k
$w(v)$, w_v	Laufzeit des Tasks v $(= \sum_k y_{v,k} w_{v,k})$
$w^{\min}$	Breite des Streifens im 2D-SPP, d.h. Laufzeit des Taskgraphen auf dem gegebenen System.
$W^{\max}$	Obere Schranke für die Breite des Streifens im 2D-SPP und somit obere Schranke für die Laufzeit des Taskgraphen
x_i	Startzeit des Tasks i
$y_{i,k}$	Zuordnung des Tasks i zum Prozessor k
y_i	Zuordnung des Tasks i zum y_i-ten Prozessor $(= \sum_k k y_{i,k})$

1 Motivation

Die Anforderung an die Rechenleistung eingebetteter Systeme steigt kontinuierlich. Dies ist zum einen durch die steigende Anzahl von parallel auszuführenden Applikationen, zum anderen durch die gestiegene Komplexität der Applikationen selbst bedingt. Anhand der Entwicklung des Mobiltelefons lässt sich dies sehr gut verdeutlichen. Stand zunächst der Sprachdienst im Vordergrund, erhalten immer mehr Funktionalitäten wie Radio, Kamera, Internet etc. Einzug in moderne Smartphones. Darüber hinaus ist mit dem Bedarf an größeren Datenraten auch die Komplexität der einzelnen Applikationen (von GSM über UMTS zu LTE, H.263 zu H.264, etc.) deutlich gestiegen.

Um diesen gestiegenen Bedarf an Rechenleistung nachzukommen, verzeichnet sich sowohl im General Purpose Computing als auch bei eingebetteten Systemen ein Trend zu Multi-Processor-Systems-on-Chip (MPSoC) [31]. Der Prozessor wird zum NAND-Gate der Zukunft [64]. Die Möglichkeit der Wiederverwertung einzelner Komponenten analog zum Baukastenprinzip verringert dabei die Designkomplexität und die Zeit bis zur Marktreife konkreter Systeme. Zudem weisen MPSoCs durch die Programmierbarkeit einzelner Prozessoren eine höhere Flexibilität als applikationsspezifische integrierte Schaltkreise auf.

Beim Entwurf eingebetteter Systeme sind die Applikationen und deren Verhalten untereinander zur Entwurfszeit bekannt, wobei Änderungen im Laufe der Lebenszeit des Systems durch Softwareupdates möglich sind. Die Qualität eines MPSoCs ist daher nicht nur von statischen Parametern abhängig, sondern auch bzw. insbesondere von der Eignung des Systems für das jeweilige Applikationsszenario. Hieraus ergeben sich zwei zentrale Fragestellungen:

1. *Welche und wie viele Komponenten sind notwendig, um die aus dem Applikationsszenario resultierenden Anforderungen zu erfüllen?*

2. *Wie kann das gegebene Applikationsszenario sinnvoll auf die parallele Architektur abgebildet werden?*

Die erste Frage ist dadurch motiviert, dass insbesondere in mobilen Geräten der Energiebedarf von entscheidender Bedeutung ist. Daher sollte das System selbst derart

dimensioniert sein, dass es die durch das Applikationsszenario definierten Anforderungen – wie bspw. die Einhaltung von Zeitschranken (Deadlines) – bei möglichst geringem Energiebedarf erfüllt. Das „je mehr, desto besser“-Paradigma des General Purpose Computing ist hier daher nicht anwendbar. Durch die gegenläufigen Optimierungskriterien wie bspw. Leistung und Energiebedarf ist vielmehr die Optimierung des Systems auf das jeweilige Applikationsszenario erforderlich. Des Weiteren erfordert die Parallelität der Architektur eine Aufteilung der Applikationen bzw. deren Teilaufgaben auf die einzelnen Prozessoren. Dieses Problem des Schedulings unter Ressourcenbeschränkungen wird als Mapping bezeichnet.

Die aufgeworfenen Fragen sind für Applikationsszenarien mit geringer Komplexität und daraus resultierend geringer Anzahl von Alternativen für das System von erfahrenen System-Designern durch die „Methode des scharfen Hinsehens“ lösbar. Mit der steigenden Komplexität der Applikationen selbst sowie der steigenden Anzahl parallel auszuführender Applikationen werden zukünftige Systeme jedoch aus mehreren hundert Prozessoren bestehen – von General Purpose Prozessoren bis zu anwendungsspezifischen Hardware-Beschleunigern [8]. Damit einhergehend wächst die Anzahl der zu betrachtenden Systeme, d.h. die Größe des Entwurfsraums, exponentiell. Daher werden Methoden zur schnellen Evaluation eines einzelnen Systems benötigt, welche die Exploration des Entwurfsraums (DSE = Design Space Exploration) erlauben.

In den folgenden Kapiteln wird ein Ansatz vorgestellt, der durch schnelle Evaluation die Exploration des Entwurfsraums, also die Beantwortung der ersten Frage, ermöglicht. Im Gegensatz zu bekannten Ansätzen wird dabei zunächst nicht auf die Optimierung des Mappings und dessen Analyse zurückgegriffen, d.h. die zweite Frage zunächst ignoriert. Da dieses Problem des Mappings NP vollständig ist [54], ermöglicht die Umgehung dieses Problems in erster Instanz die globale Exploration auch großer Entwurfsräume.

Erst in der lokalen Optimierung der in der globalen Exploration identifizierten Kandidatensysteme wird ein Mapping auf eben diese vorgenommen. Hierbei wird sowohl auf exakte Verfahren unter Verwendung linearer Programmierung als auch aufwändiger Heuristiken verzichtet. Stattdessen wird auf eine einfache Heuristik zurückgegriffen. Es wird gezeigt, dass diese im Vergleich zu exakten Verfahren gute Ergebnisse bei geringer Komplexität und somit kurzer Laufzeit erzielt. Darüber hinaus unterstützt die vorgestellte Methode komplexe Applikationsszenarien, in denen – anders als in vielen bekannten Ansätzen zur Entwurfsraumexploration – nicht nur eine einzelne Applikation, sondern mehrere nebenläufige, zu beliebigen Zeiten startende Applikationen analysiert werden können.

Im Folgenden werden aufgrund der Nähe des Problems zum Compilerbau Begrifflichkeiten aus diesem Bereich verwendet und als bekannt vorausgesetzt. Für eine Einführung sei auf [32] verwiesen. Aus dem gleichen Grund sind Grundkenntnisse der

linearen Optimierung [50] und der Graphentheorie [24] hilfreich. Im Kapitel 2 wird zunächst ein Überblick über die Entwurfsraumexploration gegeben, wobei Grundlagen, bestehende Ansätze sowie der in dieser Arbeit entwickelte Ansatz kurz erläutert werden. Darauf aufbauend wird in den sich anschließenden Kapiteln auf die Details des Ansatzes näher eingegangen. Für eine Übersicht sei auf das Inhaltsverzeichnis verwiesen.

2 Entwurfsraumexploration

Ziel der Entwurfsraumexploration – ob automatisch oder manuell – ist die Dimensionierung eines MPSoCs für ein gegebenes Applikationsszenario sowie die Abbildung des Szenarios auf dieses System. Da der Entwurfsraum für Applikationsszenarien geringer Komplexität wie bspw. beim GSM Standard überschaubar ist, kann in diesem Fall eine manuelle Exploration durchgeführt werden. Dies entspricht dem derzeitigen Vorgehen der manuellen Abbildung auf eine gewählte Architektur und Optimierung des Systems durch die Analyse von Simulationsergebnissen. Mit steigender Komplexität des Applikationsszenarios – wie bspw. beim Übergang von GSM über UMTS hin zu LTE zu beobachten – steigt die Größe des Entwurfsraums jedoch exponentiell, so dass die manuelle Exploration aus Zeitgründen nicht mehr durchführbar ist. Insbesondere die Iteration über manuelle Abbildungen des Applikationsszenarios mit anschließender Simulation auf einem gegebenen System resultiert dabei in einem hohen Zeitaufwand für die Bewertung eines einzelnen Systems. Daher wird eine Methode benötigt, die eine schnelle Bewertung eines einzelnen Systems erlaubt und somit die Betrachtung einer Vielzahl von Systemen ermöglicht.

Im Entwurfsprozess ersetzt die automatische DSE dabei nicht bereits bestehende Methoden auf den jeweiligen Abstraktionsebenen. Vielmehr dient sie zur Reduktion des Entwurfsraums auf eine kleine Anzahl sinnvoll erscheinender Kandidatensysteme, die wiederum mit bestehenden Methoden analysiert und verfeinert werden (Abb. 2.1). Hieraus folgt implizit, dass für die Exploration eine über bisherige Ansätze hinausgehende Abstraktion des Problems erfolgen muss. Da die Exploration Teil des Entwurfsprozesses ist, ist es sinnvoll, wenn die bestimmten Kandidaten als Ergebnis der DSE nahtlos an die nächste Abstraktionsebene übergeben werden können. Die Exploration ist somit als Compiler eines plattformunabhängigen Modells (nämlich des Applikationsszenarios) in ein plattformspezifisches Modell zu verstehen.

Bei der Exploration selbst sind dabei mehrere Kriterien zu betrachten. So ist neben der Einhaltung der Designvorgaben die Optimierung bspw. hinsichtlich Leistung, Energieverbrauch, Programmierbarkeit, Flexibilität, Fläche, Entwurfszeit und anderen Kriterien erforderlich [36, 60]. Leistung ist hier hinsichtlich Durchsatz und Latenz der einzelnen Applikationen zu verstehen. Bei Applikationsszenarien, in denen für die

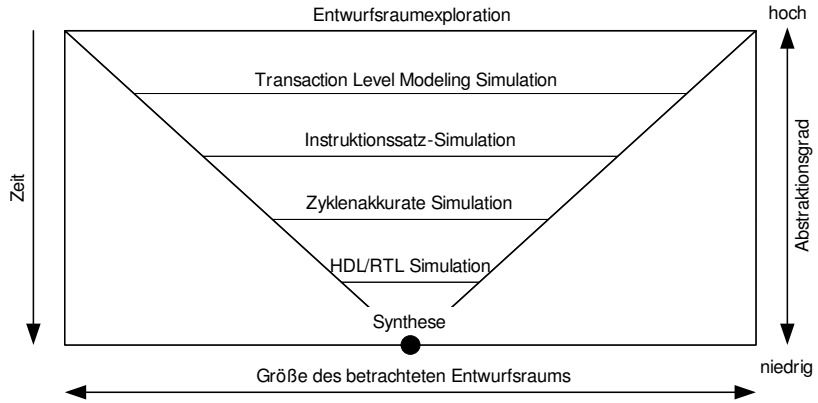

Abbildung 2.1: Einordnung der Entwurfsraumexploration im Entwurfsprozess angelehnt an [36]

einzelnen Applikationen Deadlines existieren, wird die Einhaltung der Latenz vorausgesetzt und rückt daher als Optimierungskriterium in den Hintergrund. Entwurfszeit berücksichtigt die Komplexität des Designs und beeinflusst damit die Zeit des Produkts bis zur Marktreife.

Insgesamt ist bei den Kriterien eine starke Ähnlichkeit zu denen der Implementierung einzelner Signalverarbeitungsalgorithmen in Hardware [30] festzustellen. Bei MPSoCs rückt das Kriterium der einfachen Programmierbarkeit jedoch stärker in den Vordergrund. Diese ist bei applikationsspezifischen integrierten Schaltkreise (ASICs = Application Specific Integrated Circuits) nicht erforderlich und bei einzelnen Prozessoren i.A. gegeben. Für MPSoCs ist diese jedoch keinesfalls selbstverständlich [60]. Flexibilität wird bei ASICs üblicherweise mit dem Zeitaufwand für Änderungen an der Hardware in Verbindung gebracht. Im Kontext der Multi-Prozessor-Systeme ist Flexibilität hinsichtlich der Anpassungsfähigkeit auf Änderungen in den durch Standards definierten Applikationen zu verstehen. Dies beinhaltet sowohl den Aufwand, der mit Änderungen des Systems selbst verbunden ist, als auch Änderungen, die ohne Änderungen des Systems selbst durch Softwareanpassungen durchgeführt werden können. Letztere Möglichkeit ist ein entscheidender Vorteil von MPSoCs gegenüber ASICs und Wegbereiter für Software Defined Radio (SDR) [6].

Betrachtet man die einzelnen Kenngrößen, so sind einige hiervon, wie bspw. Fläche oder Flexibilität für eine gegebene Architektur, unabhängig vom jeweiligen Applikationsszenario. Andere hingegen wie Leistung oder Energiebedarf hängen zusätzlich vom Verhalten der einzelnen Applikationen auf dem System ab. Bei der manuellen Exploration bspw. mit Modellen auf Transaktionsebene (TLM = Transaction Level

Modeling) wird daher auf eine konkrete Abbildung des Applikationsszenarios mit anschließender Simulation zur Analyse des Verhaltens auf dem System zurückgegriffen. Dass diese konkrete Abbildung zunächst nicht notwendig ist, wird in Abschnitt 2.2 erläutert, in dem auch ein Überblick über den entwickelten Ansatz gegeben wird. Vorher soll jedoch auf bekannte Arbeiten auf diesem Gebiet eingegangen werden.

2.1 Bestehende Ansätze

Für die strukturierte Entwurfsraumexploration hat sich der Y-Chart-Ansatz [46] durchgesetzt (Abb. 2.2). In diesem wird in der Modellierung zunächst die strikte Trennung des Applikationsszenarios von der Architektur vorgenommen. Das plattformunabhängige Modell des Applikationsszenarios wird dann auf die zu betrachtende Architektur abgebildet. Durch die Analyse des resultierenden Mappings werden im Anschluss die Kennzahlen des Systems bei gegebenem System, Applikationsszenario und Mapping ermittelt. In einer Feedbackschleife können dann sowohl Mapping als auch Architektur und Applikationsszenario selbst modifiziert werden, um eine Verbesserung einzelner Kennzahlen zu erreichen. Das Mapping bildet somit den zentralen Kern der Methode, auch wenn es in erster Instanz lediglich ein Hilfsmittel ist, um die Bewertung einer Architektur zu ermöglichen.

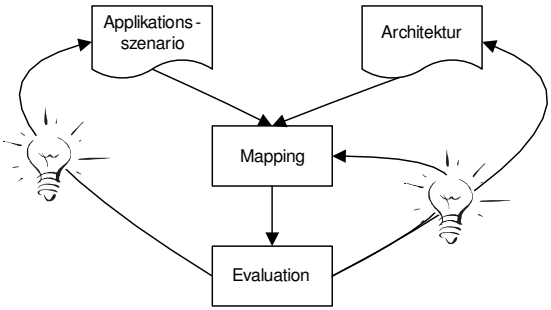

Abbildung 2.2: Y-Chart-Ansatz zur Entwurfsraumexploration nach [46]

Der beim Y-Chart-Ansatz betrachtete Entwurfsraum wird über die drei Dimensionen Applikation, Architektur und Mapping aufgespannt (Abb. 2.3). Aus der Illustration des Y-Chart-Ansatzes wird jedoch nicht deutlich, welche Feedback-Schleife wie oft zu nehmen ist. Anders gesagt ist die Größe der einzelnen Dimensionen nicht definiert. Im Bereich eingebetteter Systeme ist das Modell des Applikationsszenarios in den meisten Fällen und bei geeigneter Wahl des Models of Computation vorgegeben. Änderungen des Applikationsszenarios bedeuten daher lediglich die Betrachtung

alternativer Umsetzungen der einzelnen Applikationen. Die Blockdiagramme auf Systemebene ändern sich dabei meist nur leicht bis gar nicht. Die Anzahl zu betrachtender Applikationsszenarien ist daher eher gering. Ein weiteres Detail, dass im Y-Chart selbst nicht dargestellt wird, ist die enge Verzahnung von Applikation, Architektur sowie Mapping. Sinnvolle Prozessortypen sowie Mappings definieren sich oft zu großen Teilen durch das Applikationsszenario selbst. Aufgrund dieser vom Systemdesigner bereitgestellten Seiteninformationen kann der Entwurfsraum ggf. erheblich reduziert werden und zu Vorteilen gegenüber Ansätzen führen, die auf „blinder“ Suche basieren.

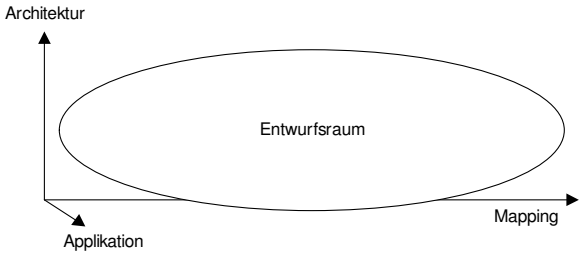

Abbildung 2.3: Die Dimensionen des Entwurfsraums bei Verwendung des Y-Chart-Ansatzes

Prinzipiell ist der Y-Chart-Ansatz selbst zunächst unabhängig von der Abstraktionsebene. Um die Iteration über verschiedene Konfigurationen zu ermöglichen, qualifizieren sich Modelle auf niedrigeren Abstraktionsebenen durch den hohen Zeitaufwand einer einzelnen Iteration nicht für diese Methode [63].

Alle den Y-Chart-Ansatz umsetzende Methoden eint hinsichtlich der Modellierung des Applikationsszenarios die Unterteilung der einzelnen Applikationen in Funktionen, um die Verteilung der Applikation auf mehrere Prozessoren zu ermöglichen. Dieses Vorgehen entspricht den Grundsätzen guter Programmierung selbst in sequentiellen Sprachen. Insgesamt ist dieses Konzept eine Umsetzung von Blockdiagrammen, wobei die Blöcke und die durch Datenabhängigkeiten definierten Kanten des Blockdiagramms einen Graphen beschreiben. Abhängig von der gewählten Beschreibungssprache ist dieser Graph dabei entweder explizit oder implizit gegeben. Auf diesen Grundgedanken wird in Kapitel 3 genauer eingegangen.

Im weiteren Verlauf dieses Kapitels sollen vorrangig bestehende Methoden zur Evaluation und Ansätze zur Suche im Entwurfsraum vorgestellt werden. Ansätze des automatischen Mappings werden in Kapitel 5 behandelt. Der Fokus liegt hier auf der kurzen Beschreibung ausgewählter Ansätze, um die entwickelte Methode abzugrenzen. Für eine ausführlichere Darstellung sei auf die exzellente Übersicht von Matthias Gries [36] verwiesen.

2.1.1 Methoden zur Evaluation

Die Evaluation kann entweder simulationsbasiert oder analytisch erfolgen. Ein simulationsbasierter Ansatz, dessen Simulationsmodell den TLM-2.0 Standard [65] umsetzt, wird in [45] beschrieben. Die Simulation erfolgt dabei auf virtuellen Prozessoren, auf die die einzelnen Funktionen manuell abgebildet werden müssen. Die Verwendung virtueller Prozessoren ermöglicht dabei die Simulation vor der Umsetzung des eigentlichen Prozessors. Dieser Ansatz erlaubt zudem die Vermischung verschiedener Abstraktionsebenen. Für vorhandene Prozessormodelle kann bei Bedarf ein Instruktionssatzsimulator (ISS) anstelle des virtuellen Prozessors eingebunden werden, wohingegen für (noch) nicht existierende Prozessormodelle die virtuellen Prozessoren inkl. deren Abschätzung der relevanten Kennzahlen im Modell verbleiben. Diese an der RWTH Aachen entwickelte Methodik ist als Virtual Processing Unit (VPU) kommerziell in CoWares Platform Architect [21] verfügbar. Eine Abstraktionsebene höher ist das MAPS Projekt ebenfalls der RWTH Aachen einzuordnen. Hierbei wird zunächst sequentieller C Code vom Benutzer geleitet in parallele Threads umgewandelt [14]. Das Verhalten der so generierten Threads kann dann über die manuelle Zuordnung auf virtuelle Prozessoren analysiert werden [15].

Andere simulationsbasierte Ansätze wie bspw. die Sesame Explorationsumgebung [71] als Teil des integrierten Daedalus Entwurfsansatzes [66] oder das MESCAL-Framework [63] nutzen graphenbasierte Beschreibungen der Applikationen und Architekturen. In einem expliziten Mapping Layer können die räumliche Zuordnung der Funktionen auf Prozessoren sowie die zu verwendenden Kommunikationsprotokolle variiert werden. Beide Applikationen unterstützen dabei unterschiedliche Models of Computation (MoC), wobei sich MESCAL der vielfältigen Möglichkeiten der Ptolemy-Umgebung [27] bedient. Ebenso ist in beiden Ansätzen eine schrittweise Verfeinerung der Architektur und des Applikationsszenarios möglich.

Simulationsbasierte Ansätze einen jedoch zwei inhärente Probleme. Zum einen setzen sie eine bestehende Implementierung voraus, zum anderen ist die Identifizierung von Corner Cases nicht trivial. Nimmt man an, dass zu Beginn des Entwurfsprozesses zunächst Blockdiagramme sowie erste Abschätzungen der Laufzeit der Funktionen auf verschiedenen Prozessortypen existieren, so ist die Implementierung eines Referenzmodells selbst in höheren Sprachen wie C oder Matlab zeitintensiv. Falls zudem die Applikation unterschiedliche Pfade nehmen kann (gegeben durch `SWITCH`/`CASE`- oder `IF`/`THEN`/`ELSE`-Anweisungen), deren Auswahl von den Eingangsdaten abhängt, so ist die Identifikation der möglichen Kombinationen aufwändig. Da die Ausführungszeit je nach ausgewähltem Pfad jedoch stark variieren kann, ist diese Identifikation notwendig. Hinzu kommt der zeitliche Aufwand der Simulation selbst. Insgesamt sind simulationsbasierte Ansätze daher eher dafür geeignet, bereits als sinnvoll erachtete Systeme weiter zu testen.

Da in analytischen Ansätzen keine Ausführung der Applikationen erfolgt, kön-

nen hier alle Pfade gleichzeitig analysiert werden. Durch den Verzicht auf die Simulation kann die Exploration zudem vor der Implementierung erfolgen. Um eine Evaluation eines Systems zu erlauben, wird dabei auf Abschätzungen der einzelnen Funktionen zurückgegriffen. Bei der analytischen Exploration bedient man sich häufig diverser Optimierungsalgorithmen. In [78] wird bspw. ein Ansatz vorgestellt, in dem das gesamte Problem als gemischt ganzzahliges lineares Optimierungsproblem (MILP = Mixed Integer Linear Program) beschrieben wird. Hier wird anstelle der durch die Simulation zur Laufzeit bestimmten Kennzahlen der einzelnen Funktionen auf Abschätzungen derselben zurückgegriffen. Durch die Komplexität des Problems sowie die schlechte Skalierung sind diese Methode wie auch andere MILP benutzende Ansätze jedoch nur auf kleinere Probleme anwendbar.

Einen anderen Weg zeigt die an der ETH Zürich entwickelte Modular Performance Analysis [16] auf. Hier wird zunächst lediglich die Architektur durch einen Graphen beschrieben. Für die Analyse des Verhaltens des Applikationsszenarios auf der Architektur werden nun im Real-Time Calculus beschriebene [85] Arrival sowie Service Curves genutzt. Die Arrival Curves beschreiben Event Streams und geben die Anzahl der zu erwartenden Ereignisse (d.h. auszuführende Funktionen) für die jeweiligen Prozessoren über die Zeit an, die wiederum vom Mapping abhängen. Durch die Verwendung von Upper und Lower Arrival Curves können dabei bspw. Jitter periodischer Events sowie das Verhalten sporadischer Events analysiert werden. Hierzu werden Upper und Lower Service Curves benutzt, die minimal und maximal bearbeitbare Events der jeweiligen Ressourcen im System beschreiben. Ein Problem dieses Ansatzes liegt jedoch in der Bestimmung der Arrival Curves. Können unabhängige, periodische Events gut abgebildet werden, führen sporadische Events sowie untereinander abhängige Events zu stark abweichenden Upper und Lower Service Curves und somit ggf. zu einer Überdimensionierung des Systems. Auch bleibt die Verteilung innerhalb der durch die beiden Arrival Curves definierten Spanne unberücksichtigt. Der Ansatz ist daher eher für das Design von Systemen mit harten Echtzeitanforderungen geeignet.

Insgesamt ist festzustellen, dass sowohl simulationsbasierte als auch sich linearer Optimierung bedienende analytische Ansätze die Exploration großer Entwurfsräume verbieten, da die Exploration zu zeitintensiv ist. Andere analytische Ansätze vereinfachen das Problem zu stark, als dass diese für Systeme mit weichen Echtzeitanforderungen hinreichend gute Ergebnisse liefern. In Abschnitt 2.2 wird daher ein Ansatz vorgestellt, der durch die Verwendung von Heuristiken und moderater Abstraktion des Problems gegenüber simulationsbasierten Ansätzen die globale Exploration großer Entwurfsräume ermöglicht.

2.1.2 Strategien der Exploration

Die möglichen Strategien zur Exploration des Entwurfsraums lassen sich grundsätzlich in drei Fälle aufteilen: Erschöpfende Suche, zufällige Auswahl von Designpunkten und geleitete Suche. Bedingt durch die Größe des Entwurfsraums erfordert die erschöpfende Suche eine schnelle Evaluation eines Systems. Da der Entwurfsraum mit steigender Problemgröße jedoch exponentiell wächst, ist dieses Vorgehen nur bedingt zu empfehlen. Ebenso wäre es sträflich, das Know-How des erfahrenen Systemdesigners zu ignorieren und rein zufällig Punkte im Entwurfsraum auszuwählen, ohne diese weiter zu verfeinern.

Genetische Algorithmen nutzen eine Mischung aus zufälliger Auswahl und geleiteter Suche. Ersteres wird dabei analog zur biologischen Evolution als Mutation, letzteres als Variation bezeichnet. In einem Selektionsschritt werden die so bestimmten Punkte im Entwurfsraum mit den bisherigen Punkten verglichen und die Population für die nächste Evolutionsstufe bestimmt. Ein bekannter, dieses Vorgehen umsetzender Algorithmus ist der Strength Pareto Evolutionary Algorithm [89]. Ein dieses Konzept nutzender Ansatz ist bspw. in das Sesame Framework integriert, in dem über die einzelnen Zuordnungen von Funktionen zu Prozessoren iteriert wird [28]. Die Qualität genetischer Algorithmen wird dabei jedoch auch entscheidend durch die Ausgestaltung der Strategien zur Mutation und Variation bestimmt, die jeweils abhängig von dem betrachteten Problem sind. Schlechte Implementierungen können daher schnell zur erschöpfenden Suche führen.

Wie der Name Strength Pareto Evolutionary andeutet, können bei genetischen Algorithmen mehrere Zielfunktionen betrachtet werden. Diese multikriterielle Optimierung ist jedoch kritisch zu betrachten. Als Ergebnis dieses Ansatzes erhält man eine Pareto-optimale Front aus nichtdominierten Systemen, in denen kein Optimierungskriterium verbessert werden kann, ohne wenigstens ein anderes zu verschlechtern. Im Gegensatz zur Optimierung nach einem Kriterium können dadurch im Verlauf des Optimierungsprozesses – insbesondere bei wie in diesem Fall vorliegenden gegensätzlichen Optimierungskriterien wie Leistung und Energiebedarf – weniger Lösungen ausgeschlossen, d.h. Bereiche des Entwurfsraums abgeschnitten werden. Hierdurch erhöht sich die Laufzeit des Algorithmus. Zudem gestaltet sich die Auswahl einer Lösung aus einer mehrdimensionalen Pareto-optimalen Front mit Dimension größer drei nicht nur durch die beschränkte Möglichkeit der Darstellung als herausfordernd. So werden bspw. im Explorationsprozess der SystemC-basierten Entwurfsumgebung SystemCoDesigner [40] für den dort durch 7.600 Punkte definierten Entwurfsraum des Motion JPEG Decoders innerhalb von 2,5 Tagen 366 nichtdominierte Lösungen berechnet. Im schlimmsten Fall wird daher das Problem der Exploration durch das der Auswahl ersetzt. Aus diesen Gründen wird die Verwendung einer lexikographischen Zielfunktion empfohlen. Bei diesem Ansatz erfolgt die Optimierung zunächst nach einem Kriterium, deren Ergebnis alle Lösungen innerhalb eines zu definieren-

den Abstands zum Optimum liefert. Die Lösungen werden nun als gültiger Raum für die Optimierung des zweitwichtigsten Kriteriums definiert usw. Dies bildet somit eine Auswahl nach, wie sie aus einer Pareto-optimalen Front erfolgen könnte.

2.2 Ansatz zur heuristischen Exploration

Im vorangegangenen Abschnitt wurde verdeutlicht, dass bestehende Ansätze zur Bewertung eines Systems ein Mapping der Applikationen auf die Architektur voraussetzen, auch wenn sie sich im Mappingansatz unterscheiden. Dies ist jedoch in erster Instanz für eine grobe Abschätzung der Leistung nicht zwingend erforderlich, was an einem kleinen Beispiel verdeutlicht werden kann. Angenommen es sollen 10 Tische gefertigt werden. Weiter sei angenommen, dass ein Tischler einen Tag für die Produktion benötigt und 5 Tischler verfügbar sind. Für die Abschätzung der Zeit, die die Tischler benötigen, um alle Tische herzustellen, oder wie die Tischler ausgelastet sein werden, bedarf es keiner großen Rechenkunst. Insbesondere ist es für die Berechnung irrelevant, welcher Tischler zu welcher Zeit welchen Tisch produziert, also über alle denkbaren 5^{10} Zuordnungen von Tischen zu Tischlern und für jede dieser Zuordnungen über die bis zu $10! \approx 3{,}6$ Mio. verschiedenen zeitlichen Ordnungen zu iterieren. Genau diese, wenn auch aufgrund der Komplexität nicht vollständige, Iteration geschieht aber bei der Bewertung eines Systems anhand eines Mappings im klassischen Y-Chart-Ansatz. Für eine erste Abschätzung reicht jedoch die Kenntnis über die vorhandenen Ressourcen sowie die inhärente Parallelität der Aufgaben aus, um das Problem zu analysieren. In Kapitel 4 wird dieser Grundgedanke erweitert, um eine globale Exploration über die Analyse der inhärenten Parallelität eines Applikationsszenarios zu ermöglichen.

Da mit der Abstraktion ggf. Ungenauigkeiten in der Abschätzung verbunden sind, wird danach auf den bekannten Ansatz des Mappings zur weiteren Analyse und der damit verbundenen Möglichkeit der lokalen Optimierung zurückgegriffen. Hierzu wird in Kapitel 5 das Problem des Mappings klassifiziert, bevor in den folgenden Kapiteln Methoden zum Mapping vorgestellt werden. Hierbei werden zur Verfügung stehende Seiteninformationen des Systemdesigners genutzt, um die Anzahl der zu betrachtenden Mappings weiter zu reduzieren. Ein Vergleich eines exakten Verfahrens mit der einfachen Heuristik des listenbasierten Mappings in Kapitel 6 wird zeigen, dass das listenbasierte Mapping vergleichsweise gute Ergebnisse liefert. Insbesondere ist festzustellen, dass beim listenbasierten Ansatz lediglich ein Mapping aufgestellt wird. Die Iteration über verschiedene Mappings ist daher nicht notwendig, wenn ein geringer Qualitätsverlust toleriert werden kann. Hier sei angemerkt, dass in der Vergangenheit die Einführung neuer Abstraktionsebenen jeweils zu einem solchen Verlust an Genauigkeit geführt hat, was in einer gewissen Überdimensionierung des Systems resultiert [82].

Abbildung 2.4 zeigt einen Überblick über den gesamten Prozess der DSE. Sowohl für die globale Exploration als auch für die lokale Optimierung wird auf abstrakte Graphen des Applikationsszenarios und der Architektur zurückgegriffen. Für die Modellierung einer Applikation werden gerichtete, azyklische Taskgraphen verwendet. Das Verhalten der Applikationen untereinander wird über Zustandsautomaten abgebildet. Details hierzu finden sich in Kapitel 3.

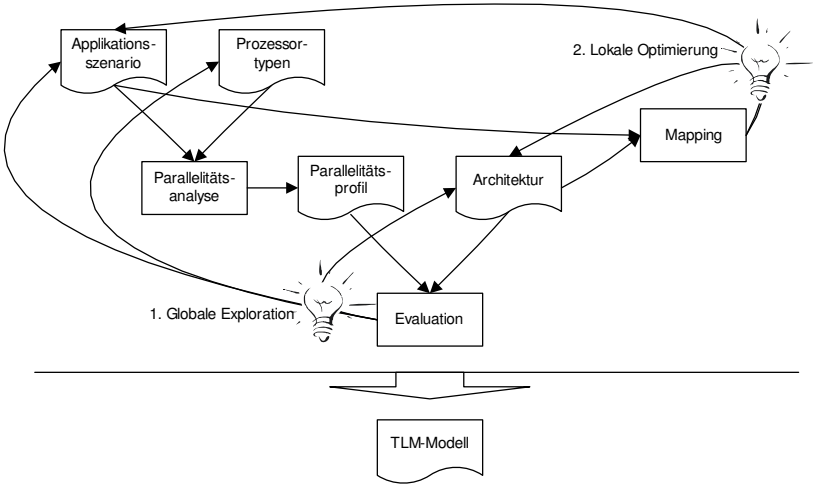

Abbildung 2.4: Überblick über den vorgestellten Ansatz zur Entwurfsraumexploration

Um die Exploration selbst vor der zeitaufwändigen Implementierung zu ermöglichen und somit die mehrfache Implementierung verschiedener Funktionen auf unterschiedlichen Prozessoren zu vermeiden, erfolgt die gesamte Exploration über die Analyse der Graphen ohne die Ausführung der einzelnen Funktionen. Hierzu werden Kennzahlen für die einzelnen Funktionen der Applikationen von einem erfahrenen Systemdesigner annotiert. Für weitere Details sei ebenfalls auf Kapitel 3 verwiesen. Erst nach der Exploration werden als Ergebnis ausführbare Modelle aus den abstrakten Graphen generiert. Momentan wird bereits die Generierung von TaskC [55] als auch eines mit dem ISS der RWTH Aachen entwickelten Austauschformats zum Import in CoWare Platform Architect [21] als TLM 2.0 Modell [65] unterstützt.

2.3 Zusammenfassung

Durch die stetig steigende Komplexität der Applikationen für eingebettete Systeme sowie die steigende Anzahl der zu bearbeitenden Applikationen selbst wächst die Größe

des Entwurfsraums heterogener Multi-Prozessor-Systeme exponentiell. Daher ist ein strukturierter Ansatz zur Exploration dieses Entwurfsraumes notwendig. Bestehende Ansätze bedienen sich dafür meist des Y-Chart-Ansatzes. Dabei erfolgt die Evaluation eines Systems durch Analyse eines Mappings. Da das Problem des Mappings jedoch komplex und daher zeitintensiv ist, ist dieser Ansatz zur Exploration großer Entwurfsräume nicht geeignet. In dieser Arbeit wird daher ein zweistufiger Ansatz vorgestellt, der in der globalen Exploration die Evaluation eines Systems ohne Mapping ermöglicht. Für die lokale Optimierung wird für das Mapping auf einfache Heuristiken zurückgegriffen. Eine Iteration über verschiedene Mappings ist nicht notwendig.